# Data structures and Algorithms LAB – BSDSF21

## (Morning and Afternoon)

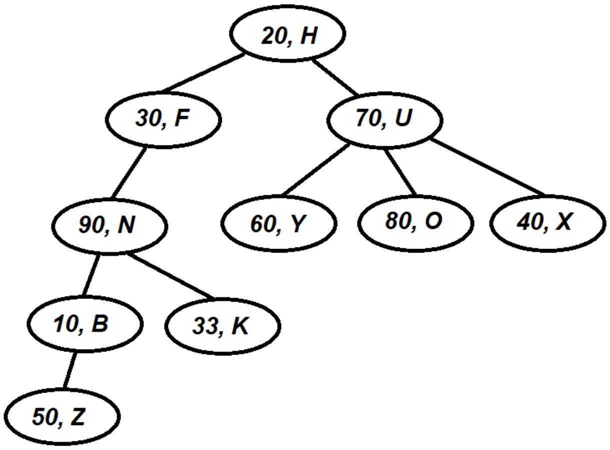
# Lab 01 – 10-01-2023

Trees are hierarchical structures as shown in diagrams below. Assuming we have a simple way to store the trees in two parallel arrays. In array **item** the objects of the tree are stored. If an object is stored at index **j** in array item, at that time in the array **pred** at index **j** contains the key of its parent object. As **root** object has no parent, so it related cell in the **pred** array is a special value. Following is ab example of a tree with all keys +ve numbers and hence -1 can be taken as special value for parent of root object.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **item** | {20, H} | {30, F} | {70, U} | {90, N} | {60, Y} | {10, B} | {80, O} | {40, X} | {50, Z} | {33, K} |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **pred** | -1 | 20 | 20 | 30 | 70 | 90 | 70 | 70 | 10 | 90 |

And its pictorial representation is:



Choose some appropriate classes or structures and create the following functions for type TREE. You have to use the concepts you learned in PF and OOP.  Do not think it as a DSA task but as a programming task. Best of Luck

1. createTree(objectCount) // create empty tree with underlying arrays and variable/objects
2. addObject(key, value, parentKey) // add an object in the tree
3. printTreeAsArrays() // print parallel arrays
4. searchParentKey(childkey) // returns the parent key for childKey
5. printChildren(parentKey) // print direct children as objects
6. countLeaves() // leaves are objects in tree with ZERO children

You also need to guess the comparative running times (normal, efficient, inefficient) of all the functions.

*******************